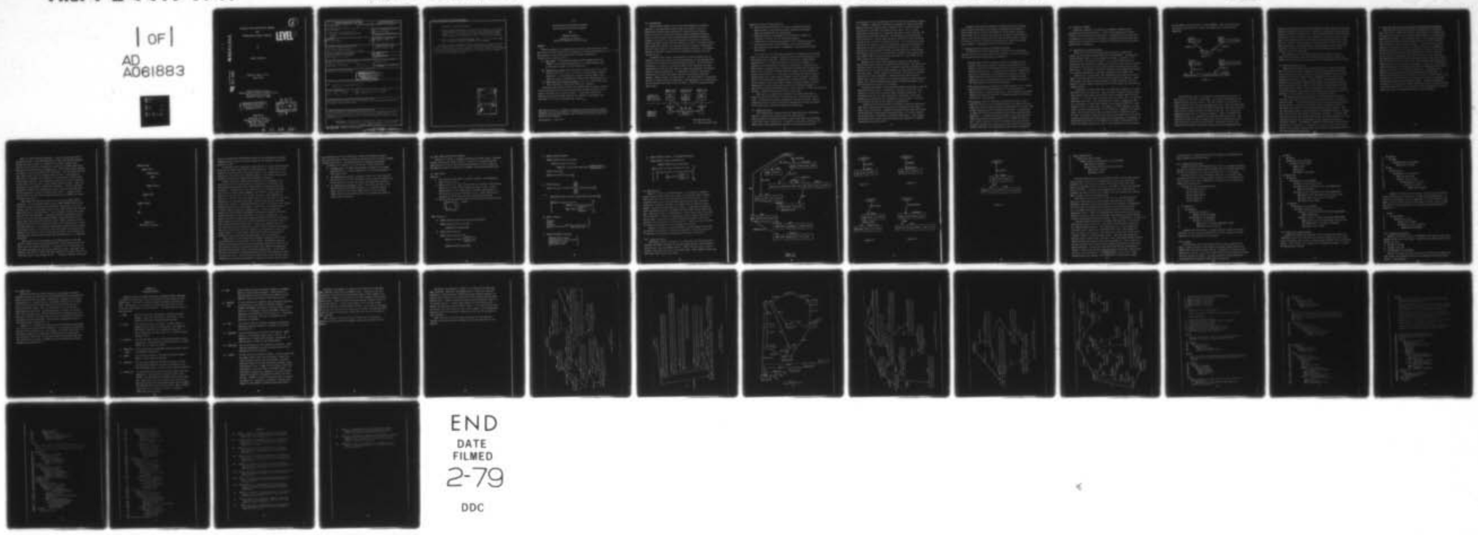
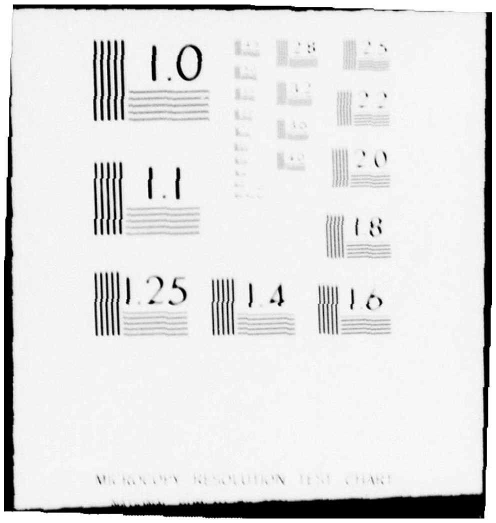
AD-A061 883   MICHIGAN UNIV ANN ARBOR DATA BASE SYSTEMS RESEARCH GROUP   F/G 9/2
AN ACCESS PATH SPECIFICATION LANGUAGE FOR RESTRUCTURING NETWORK--ETC(U)
APR 77   D SWARTWOUT   DCA100-75-C-0064
UNCLASSIFIED   TR-77-DT-7   SBIE-AD-E100 109   NL

| OF |
AD
A061883



END
DATE
FILMED
2-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART

1 B.S.

AD-E100109

AN ACCESS PATH SPECIFICATION LANGUAGE FOR RESTRUCTURING NETWORK DATABASES

LEVEL II

by

Donald Swartwout

Technical Report 77 DT 7
April 1977

Data Translation Project
Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan 48109

Supported by

Department of Defense
Defense Communications Agency
WWMCCS ADP Directorate
Reston, Virginia
DCA 100-75-C-0064

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report 77 DT 7	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 AN ACCESS PATH SPECIFICATION LANGUAGE FOR RESTRUCTURING NETWORK DATABASES		5. TYPE OF REPORT & PERIOD COVERED 9 Technical Report
		6. PERFORMING ORG. REPORT NUMBER 77 DT 7
7. AUTHOR(s) 10 Donald Swartout		8. CONTRACT OR GRANT NUMBER(s) 15 DCA 100-75-C-0064
9. PERFORMING ORGANIZATION NAME AND ADDRESS DATA BASE Systems Research Group 276 Business Administration Univ. of Michigan, Ann Arbor, Mi 48109 410 894		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 32017K, 27400, 27402
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Washington, D.C. 20305		12. REPORT DATE 11 April 77
		13. NUMBER OF PAGES 40
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Command & Control Technical Center ATTN C422, Pentagon Washington, D.C. 20301 12 43p.		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Cleared for open publication, distribution unlimited

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A 14 TR-77-DT-7

18. SUPPLEMENTARY NOTES

None 18 SBIE 19 AD-E100 109

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Translator Language Access Path Language, Data Base Restructuring, Data Translation, Network Data Bases

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Current approaches to restructuring are based on the hierarchical data model and use specific operations to achieve the required transformations. Our approach to restructuring the class of network databases has three principal characteristics:

1. Data model - The Relational Interface Model (RIM) permits the database to be viewed simultaneously as a network database and a relational

(Over) 410 894

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

68 11 03 03

database in first normal form.

2. Restructuring System which can perform the operations of the relational algebra but is not restricted to any data model dependent operations. It can perform in a simple efficient manner, user-required network transformations such as link record processing.

3. Access Path Specification Language - APSL - is a verifiably powerful, structurally simple, and descriptive language.

The access path approach permits the specification of complex restructuring transformations in terms of application-oriented concepts such as access strategies and selection criteria. A high-level Access Path Restructuring Language (APSL) based on this approach is presented, and an example of its use in restructuring is given.

Page 24 left out intentionally

-A-

An Access Path Specification Language for Restructuring Network Databases

by

Donald Swartwout

University of Michigan
Center for Database Systems Research

ABSTRACT

Current approaches to restructuring are based on the hierarchical data model and use specific operations to achieve the required transformations. The writer's ~~Our~~ approach to restructuring the class of network databases has three principal characteristics:

1) Data model - The Relational Interface Model (RIM) permits the database to be viewed simultaneously as a network database and a relational database in first normal form,

2) Restructuring System which can perform the operations of the relational algebra but is not restricted to any data model dependent operations. It can perform in a simple efficient manner, user-required network transformations such as link record processing, and

3) Access Path Specification Language - APSL - is a verifiably powerful, structurally simple, and descriptive language.

The access path approach permits the specification of complex restructuring transformations in terms of application-oriented concepts such as access strategies and selection criteria. A high-level Access Path Restructuring Language (APSL) based on this approach is presented, and an example of its use in restructuring is given.

Key Words and Phrases: database restructuring, data translation, network restructuring, network databases, restructuring software, data translation software, restructuring languages, data translation language, translation specification languages.

CR Categories: 3.70, 3.73

1.0 INTRODUCTION

The currently expanding use of very large databases in business, industrial, and governmental applications has created a need for a generalized reorganization capability for three principal reasons. First, the lack of a systematic database design methodology [DL28] results in poorly designed and consequently poorly performing databases. Secondly, the lack of an adequate requirements methodology to precisely define user needs coupled with the inability of users to formulate requirements beyond the present activities results in an outdated system design. The third major reason is the currently changing technology. New hardware advances and software capabilities are being made available at an ever increasing rate.

To address this problem, new technology is being developed for the migration of data and software between environments, or the reorganization of data within an environment [DT4,DT9,DT10,R3,R5,R9]. At the University of Michigan, a series of increasingly general data translators has been implemented over the past four years to support the migration and reorganization of data [DT1,DT3,DT11]. As indicated in Figure 1-1 the overall architecture of a data translator consists of three major functional modules - Reader, Writer, and Restructurer. The major thrust of the current version of the Michigan Data Translator is the development of a comprehensive set of reorganization capabilities for IDS/I, a DBTG-like database management system.

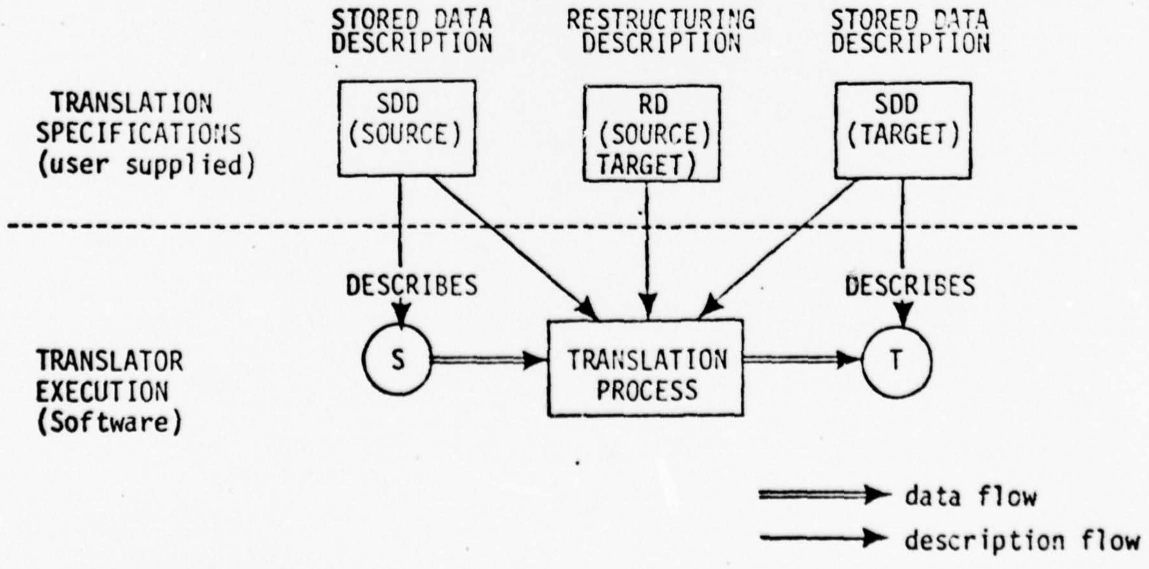


Figure 1-1

Important features of the design are:

- Complete data descriptions of the user's (currently existing) source database, and (proposed) target database are obtained using an augmented IDS data description.
- These descriptions are adequate for the Reader to create the translator's internal form of the source database.
- A high-level restructuring language has been developed to specify the restructuring of a source database into a target database.
- The execution of the process is automatic, based entirely on these descriptions.

It is the intent of this paper to present our approach to and development of a high-level language to specify restructuring transformations for the Michigan Data Translator. Basically, the goal of a restructuring transformation is to change the logical structure of a database in response to new information or processing requirements. Navathe and Fry [R2] provide a categorization of restructuring capabilities for the hierarchical class of logical structures and also develop several fundamental restructuring operations. In this paper, we broaden the scope of restructuring capabilities to the class of network logical structures including several "less theoretical" transformations which users need. To achieve this we present a data model general enough to support multiple user and implementation views; we develop an access path-oriented language for specifying restructuring transformations; and we describe a simple restructuring strategy to perform the necessary data manipulations.

This section is completed by reviewing the current approaches to restructuring and developing the requirements for a generalized restructurer. In Section 2 we discuss the approach and basic research. Section 3 presents the access path restructuring language and provides an example of its application; Section 4 contains our conclusions.

1.1 Current Research

It is interesting to observe that the large majority of the development of a restructuring technology has not taken place in the development of database management systems, but rather in the context of the data translation systems. In addition to the work at Michigan, two other efforts have addressed the problem of specifying restructuring transformations using a methodology similar to the one developed in Figure 1-1. CONVERT, a high-level restructuring language developed at the IBM Research Laboratory [R3] provides a powerful set of restructuring operations based on hierarchic data model called a Form. This is a two-dimensional representation of hierarchic data which reflects not only the schema but also the data instances. The headings to the form are the schema constructs - record types and items; relationships are maintained through the key item of the parent record type. The restructuring transformations are based on a set of Form operators which operate on one or more Forms (or their components) resulting in a new Form. With the exception of the assignment and CASE, all Form operations can be nested. An extensive repertoire of Form operators ranging from Form manipulation (e.g. MERGE, GRAFT) through built-in functions (e.g. SUM, COUNT) to the CASE statement have been developed.

Another specific data model operations approach is the work at System Development Corporation reported by Shoshani [R5]. Instead of the Form data model this approach uses a standard hierarchic data model and assumes that the source and target database have been defined in terms of this data model. Eleven "conversion operations" are defined to describe the source to target assignments. These operations range from the DIRECT function which provides a one-to-one assignment of source items to target items through INVERSION which causes the parent/dependent record type relationship operation to be inverted.

An obvious advantage of the data model operations approach is that the resultant software system is inherently simple and a set of low level subroutines can be built to perform these elementary operations. Unfortunately, the user, although involved at a high level, still views restructuring essentially as a sequence of low level steps. Thus he is not shielded from many of the details of restructuring.

In both cases the restructuring is defined in terms of a particular data model, implying that the data must be converted to this form in order to be restructured. Although the hierarchic data model facilitates the development of restructuring transformations, it is not as general as the more powerful Relational and Network data models. Thus, restructuring these more complex structures becomes very difficult in CONVERT and perhaps impossible in CDTL.

In general the low level operations approach works well on simple small logical substructures but becomes increasingly more complex in direct proportion to the size and complexity of the source structure. Furthermore, the more complex the structure, the greater the possibility that it will contain substructures that are not restructurable. Hence we found it difficult to generalize the elementary operations approach to our problem - the network class of logical structures.

1.2 Requirements for Restructuring Network Database

In reviewing the current approaches to hierarchical restructuring operations and the extended capabilities necessary for network structures we find the following requirements are necessary.

- The data model should be general enough to accommodate as many existing data models (network, relational, and hierarchic) as possible, while at the same time providing a basis for a practical restructuring algorithm. It should also provide the restructuring user with as many views of the database as possible. Further, it should provide the basis for a small set of constructs, simple enough to be easily understood by users, and readily analyzed by the implementors of the restructuring system.
- The restructuring system should be provably powerful, but it should not be limited to the implementation of a set of operations which manipulate a specific data model. Its restructuring algorithm should be as simple as possible, for obvious reasons of efficiency, verification, and debugging.
- The language in which restructuring specifications are written should be based on a set of constructs which are simple and easily understood, but also general enough to accommodate the multiple views of data supported by the restructuring data model. In particular, it should not be tied to any specific set of data-model-dependent operations. It should provide the capability to specify the less formal restructuring transformations such as adding an indexing set. Finally, the full power of the restructuring system should be readily available; all restructuring specifications should have similar structures, and essentially the same level of complexity.

2.0 TECHNICAL APPROACH

Three main components are necessary to develop a powerful user-oriented restructuring capability: a high-level language for the specification of restructuring transformations, a comprehensive data model which is supportive of the language and capable of representing established data models, and an algorithm which is capable of performing the transformations specified by the language.

2.1 The Relational Interface Model

Since none of the existing data models provide the capabilities needed, the Relational Interface Model [R7] was developed. The Relational Interface Model (RIM) is not a new model but rather a synthesis of the salient features of extant data models. It is sufficiently general to accommodate the hierarchical, network, and relational views of data necessary to represent the major data models implemented in current database management systems. Furthermore, it permits a database to be viewed simultaneously as a network database and a relational database in First Normal Form.

Record types in the network view correspond one-for-one with relation types in the relational view, and item types correspond to domains. Since First Normal Form does not permit domains to contain relations, the RIM prohibits certain network constructs such as naming groups and contained-in-repeating groups.

In addition to records and items, two additional constructs of the network model need to be addressed. Sets are used to represent logical connections between record types, and keys are used to uniquely identify record instances. In the RIM, these are implemented in a uniform way: by specially designated data items. Each record type must contain a primary key; that is, a collection of items whose combined values uniquely identify a record instance within its record type. For example, in DBTG databases, the hash field in a CALC record may serve as its primary key. However, some record types, such as link records, may not contain such a collection of items. In order to completely identify these records, the identities of the record instances which own them along certain sets must be known. For example, in the database of Figure 1-2, an instance of S-C-LINK is identified by the SS# of the STUDENT record which owns it along ENROLLED-IN, and the Course#

of the COURSE record which owns it along TAKEN-BY. Thus, the primary keys of certain record types must include information used to establish set membership.

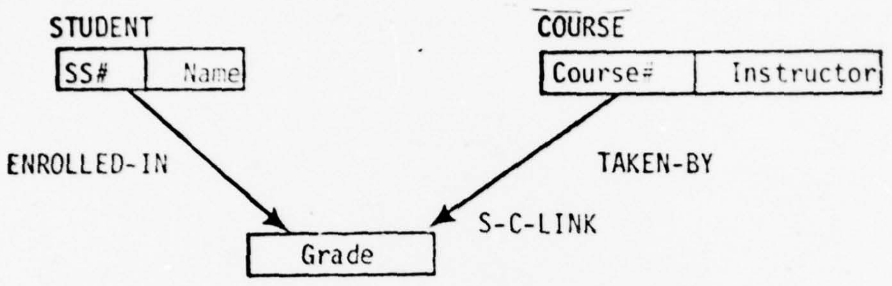


Figure 1-2

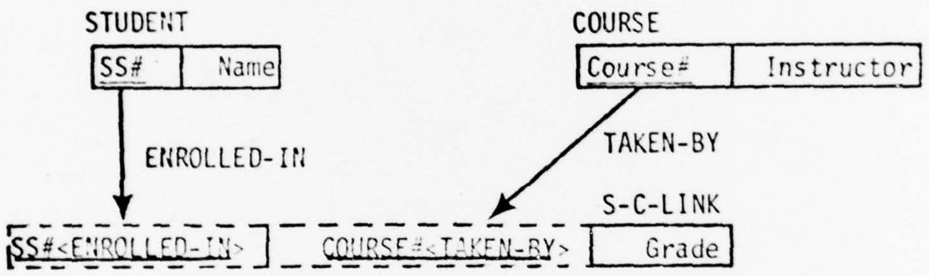


Figure 1-3

In the RIM data model, set membership is established by storing, in each instance of the member record type, a copy of the primary key of the appropriate owner record instance. This copy resides in a collection of special-purpose items, known as set-significant items. Each set-significant item corresponds to one and only one owner primary key item, and a set-significant item is used to represent one and only one set. It is said to be significant to the set it is used to represent. Figure 1-3 exhibits the database of Figure 1-2, augmented to include set-significant items. Set-significant items are shown in dotted boxes, and set arrows point to the set-significant items which represent them. Convenient names for set-significant items may be formed by concatenating the corresponding owner primary key item name with the set name enclosed in angle brackets. Primary key items are underlined. Notice that set-significant items may be primary key items as well, as in the S-C-LINK record type. Thus, during the construction of a RIM representation for a network database, the specification of primary keys for some record types is contingent upon the specification of certain set-significant items, which is, in turn, contingent upon the specification of primary keys for the owner record types. The possibility of infinite regress exists, but in our experience it has never occurred, and since IDS databases do not permit cycles of sets, it is impossible.

The identification and naming of set-significant items, and the specification of primary keys are performed within the user's source and target data descriptions, and are assumed for the restructuring specification.

2.2 The Restructuring System

Deppe [R7] has shown that any transformation of a relational database which can be specified by a sequence of the relational algebra operations Join, Selection, Projection, and Union can be performed by another sequence in which all Joins are computed first, followed by all Selections, then Projections and lastly, Unions. He also gives a simple algorithm which performs all such ordered sequences with minimal use of temporary storage, and a time bound of O(n), where n is the number of target record instances (or tuples) to be created. Thus any restructuring system which is capable of executing Deppe's algorithm has substantial power; it can perform all the operations of the relational algebra on a relational database.

The restructuring system developed for the Michigan Data Translator is based on the Deppe algorithm and views the data by using the Relational Interface Model. It can perform any restructuring transformation which can be specified by a sequence of relational algebra operations applied to the source data viewed from the relational perspective of the RIM. Thus its restructuring power is at least that of the relational algebra. Furthermore, the system is not limited to relational operations. It can be understood entirely from the network point of view, and specifications for it can be written by users with no knowledge of the relational data model or relational algebra. In addition, a synthesis of the network and relational perspectives is possible. To some extent, the user has the "best of both worlds"; he can select the data model best suited to a particular portion of his restructuring transformation.

2.3 An Access Path Specification Language for Database Restructuring

APSL, the Access Path Specification Language used by the Michigan Data Translator to encode restructuring specifications, has evolved into a language which satisfies the necessary requirements for a restructuring specification language. In its present form, it is a powerful language; the restructuring transformations which can be specified in it have at least the power of the relational algebra. It is an inherently simple language; its fundamental construct---the access path---is adequate to direct the restructuring system's retrieval of source data, but it does so without making use of any schema manipulations. That is, APSL describes the source structures from which target data may be retrieved, not operations intended to convert the source schema to the target schema. Thus it is not tied to any specific set of operations which manipulate the schema constructs of a particular data model. As such, it is more easily understood and used than restructuring languages which require the user to master a special set of restructuring operations. Furthermore, the descriptive approach taken by APSL has considerable flexibility. It permits the specifications for a complex restructuring transformation to be structured in exactly the same way as the specifications for what might be considered more elementary transformations. Since this fundamental structure is basically quite simple, we feel justified in stating that APSL is a verifiably powerful, but structurally simple, restructuring language which lays claim to a reasonable degree of user-friendliness.

APSL is a block-structured language. Figure 2-2 shows the typical nesting of blocks. At the outermost level is the TARGET RECORD statement. An APSL description (that is, a complete set of APSL statements for a particular restructuring transformation) contains one TARGET RECORD statement for each target record or relation type. Each TARGET RECORD statement is made up of one or more ACCESS PATH statements. From the network perspective, APSL assumes that each target record instance is represented by data which is contained in an instance of some hierarchical substructure of the source database. This substructure need not be a strict subschema of the source data structure; it may contain unravelled loops. These substructures are described by ACCESS PATH statements. From the relational perspective, ACCESS PATH statements are used to describe the joins which are to be computed in order to create target tuples. A target relation is the union of all the tuples created according to the ACCESS PATH specifications given for it.

Each ACCESS PATH statement consists of one or more SOURCE RECORD statements. From the network point of view, SOURCE RECORD statements describe the nodes of the hierarchies from which target data is to be retrieved; and from the relational point of view, they describe the relations which take part in join operations. At the innermost level of APSL structure are the ITEM statements. In network-oriented restructuring, their task is to specify the correspondence between target data items and the source data items which represent them, and to specify selection criteria which distinguish valid or desirable instances of a hierarchy from invalid or undesirable ones. In relational restructuring, they are used to insure that target tuples are created only from source tuples with appropriately matching join fields, to specify selection operations, and to specify the correspondence between target domains and source domains, i.e., projection operations.

Syntax details and a moderately sized example appear in the next section. Notice that we have delineated APSL's specification of the four basic relational algebra operations. It should be clear from Section 3 that they are adequate to specify any sequence of joins, selections, projections, and unions, in that order, when applied to the source data in its relational form as seen through the RIM. By Deppe's theorem, we conclude

```
TARGET RECORD
     ACCESS PATH
          SOURCE RECORD
              ITEM
                .
                .
                .
          SOURCE RECORD
             .
             .
             .
      ACCESS PATH
          .
          .
          .
TARGET RECORD
    .
    .
    .
EOF
```

Figure 2-2
APSL Block Structure

that any restructuring transformation which can be performed by relational algebra operators acting on the source data in its relational form can be specified in APSL.

In addition, APSL's descriptive power extends to certain transformations which are not readily describable in relational terms. These include such network-oriented operations as creating indexing sets, bypassing nodes on hierarchies, and altering the implementations of many-to-many relationships. Examples of the last two appear in the Example of Section 3.

Finally, practical considerations dictate that even though we have relatively little field experience in full-scale data translation, our restructuring systems must give as much thought to execution efficiency as possible. Restructuring is an inherently time-consuming process which must create a complete target database "from scratch", and in order to obtain the necessary data, must traverse the entire source database at least once. In fact, given the current state of the art, parts of the source database will be traversed many times. We have found that in some real-world cases, however, a large portion (70% or more) of the source data remains unaltered between source and target. In such cases, much processing time is spent simply copying this unchanging data. Using an interesting synthesis of the relational and network perspectives afforded by the RIM, we have developed an APSL feature known as partial restructuring. It permits the user to identify the unchanging portion (if any) of his database. Since complete representation of all the unchanged target records and the sets interrelating them already exist in the source database, they are not represented in the restructuring system's target internal form database, but are written directly from the internal source database into the user's target database. In this manner, substantial savings in restructuring processor time and internal temporary storage use can be achieved.

The synthesis of the network and relational viewpoints that we used to develop partial restructuring is very simple. The problems posed by partial restructuring are most acute with network databases,in which it can be extremely difficult to maintain the sets which connect the unchanging portion of the database to the restructured portion [R1]. In particular, the only definition of "unchanging portion" which we find adequate is stated in relational terms: those record types whose source and target RIM representations are identical are defined as unchanging records, and the sets of the unchanging portion are those which interrelate unchanging records. Thus the relational viewpoint provided by the RIM was used to solve a network restructuring problem: the identification of unchanging subnetworks.

In summary, our approach to restructuring is characterized by:

A. its data model---the RIM, which permits a database to be viewed simultaneously as a network database and a relational database in First Normal Form.

B. its restructuring system---based on a simple, efficient algorithm which can perform all the operations of the relational algebra, but is not restricted to any set of data model-dependent operations.

C. its specifications language---APSL, a verifiably powerful, structurally simple, descriptive language. It is used to specify source structures from which target data is to be retrieved, rather than operations which convert source schema structures to target schema structures.

3.0 APSL SYNTAX AND DETAILED EXAMPLES

Section 3.1 contains a complete specification of APSL syntax. Detailed semantic rules may be found in [DT12]. Section 3.2 illustrates the APSL specification of some of the restructuring operations cited in Section 2. The Appendix contains an example of a complete restructuring specification.

3.1 APSL Syntax

Notation:

1. APSL reserved words appear in capital letters; user-determined words in lower case.
2. Square brackets $\left([\quad]_m^n\right)$ indicate that the contents of the brackets must occur at least m times and no more than n times. If the upper bound is an "n" rather than an integer, the contents of the brackets may repeat arbitrarily often. Square brackets with no bounds ([ ]) indicate that their contents are optional.
3. Braces $\left(\begin{Bmatrix} \text{option}_1 \\ \text{option}_2 \\ \vdots \\ \text{option}_3 \end{Bmatrix}\right)$ indicate that exactly one of the options must be chosen.

APSL Statements:

A. TARGET RECORD Statement for Restructured Records

TARGET RECORD target-record-name

$[\text{ACCESS PATH Statement}]_1^n$

B. ACCESS PATH Statement

ACCESS PATH access-path-id

$\left[\text{target-item-name} = \begin{Bmatrix} \text{integer} \\ \text{float} \\ \text{literal} \end{Bmatrix}\right]_0^n$

$[\text{SOURCE RECORD Statement}]_1^n$

C. SOURCE RECORD Statement

```
SOURCE RECORD source-record-name
     ACCESS VIA source-set-name
           [FROM ID = parent-node-identifier [{OWNER/MEMBER} ]]
                                              {MEMBER/OWNER}
[ITEM Statement]ⁿ₀
[BLOCK ASSIGNMENT Statement]²₀
```

D. ITEM Statement

```
                                  (EQ)
                                  |NE|
source-item-name  [SELECT IF      |GT|  VALUE Statement]ⁿ₀
                                  |GE|
                                  |LE|
                                  (LT)

     [WHEN QUALIFIED BY routine-name₁[(VALUE Statement)]]ⁿ₀

     [ASSIGN TO target-item-name                          ]ⁿ
     [                [CONVERT WITH routine-name₂]        ]
     [                              (integer)             ]
     [                [NULL VALUE = {float  }   ]         ]
     [                              (literal)             ]₀
```

E. VALUE Statement

```
(integer                                              )
|float                                                |
|literal                                              |
|source-item-name [FROM source-record-name            |
(                         [ID=identifier]  ]          )
```

F. BLOCK ASSIGNMENT Statement

```
(ACTUAL DATA IN ORDER           )
|SET-SIGNIFICANT DATA BY NAME   |
|OTHER DATA BY NAME             |
(ALL DATA BY NAME               )
```

6. TARGET RECORD Statement for Unchanging Records

```
TARGET RECORD target-record-name IS

          SOURCE RECORD source-record-name

   ┌SET target-set-name IS source-set-name                    ┐n
   │    ┌target-item-name IS source-item-name                ┐n│
   │    │                     ┌integer┐ ┐                    │ │
   │    │         ┌NULL VALUE =┤float  ├ │                    │ │
   │    │         └            └literal┘ ┘                    │ │
   └    └                                                     ┘0┘0
```

3.2 APSL Examples

Figure 3-1 shows a small RIM database. It is drawn as a Bachman diagram (network view) augmented to show set-significant items in dotted boxes and primary keys, which are underlined (relational view). It is a modified subset of a larger database (see Appendix) which represents information concerning a mansion, its contents and the families living nearby. The subset we have chosen for Figure 3-1 represents neighboring houses (NEIGHBORS records), the PEOPLE who LIVE-THERE, and the AUTOs they own. Ownership of cars is implemented by a link record, since joint ownership is possible. Purely for the purposes of this example, BOSS records are included. These give the name of each person holding a managerial position, and the company he or she works for. Finally, we have the ROOMS of the mansion and the LAMPS and FURNITURE they contain.

Figures 3-2 through 3-6 show fragments of target databases which can be derived from the data of Figure 3-1 using APSL. They illustrate some of the major restructuring transformation discussed in Section 2.

3.2.1 Join, Projection

The PEOPLE relation of Figure 3-2 is the result of joining the source relations NEIGHBORS and PEOPLE on the deed number domains, then projecting onto the name, age, and address domains. Notice that this join is represented in the source data by the set LIVES-THERE. APSL statements describing the target PEOPLE record type follow.

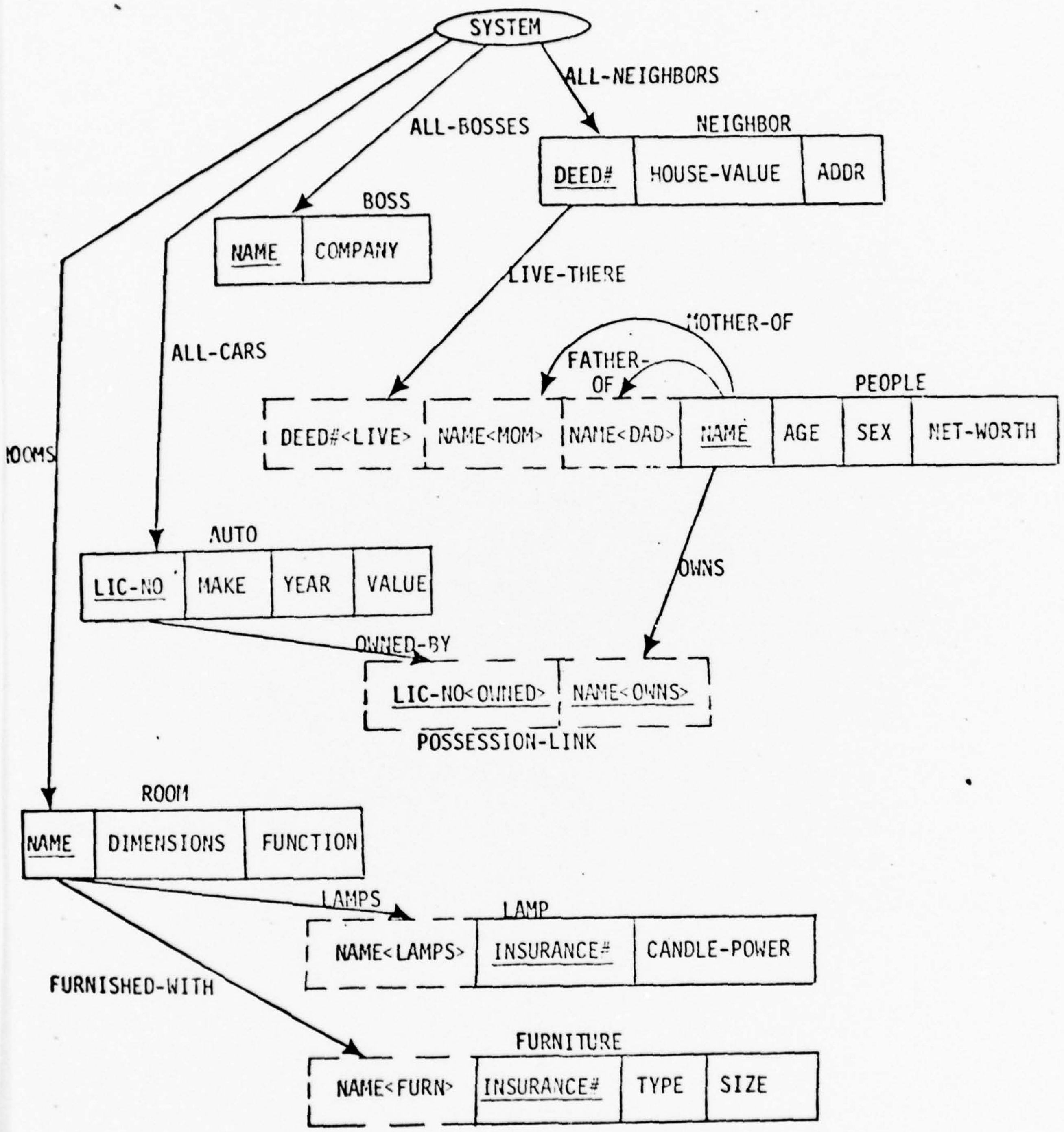


Figure 3-1
Source Data

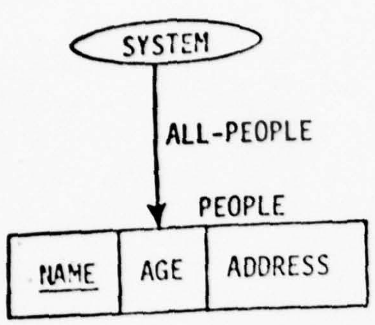


Figure 3-2

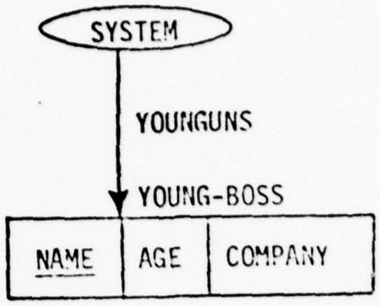


Figure 3-3

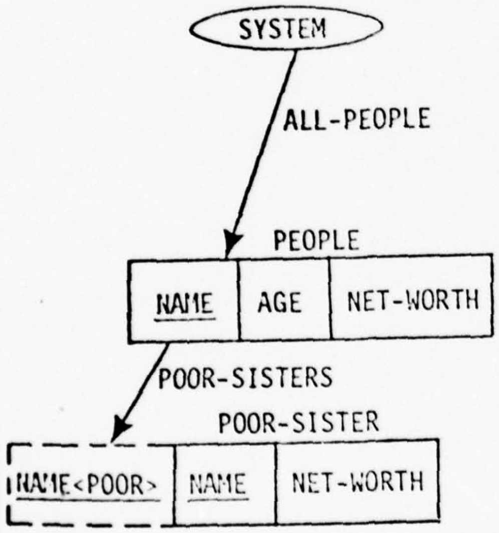


Figure 3-4

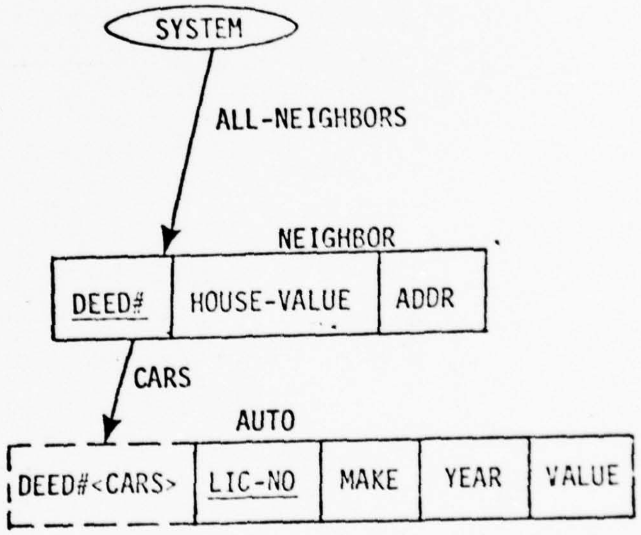


Figure 3-5

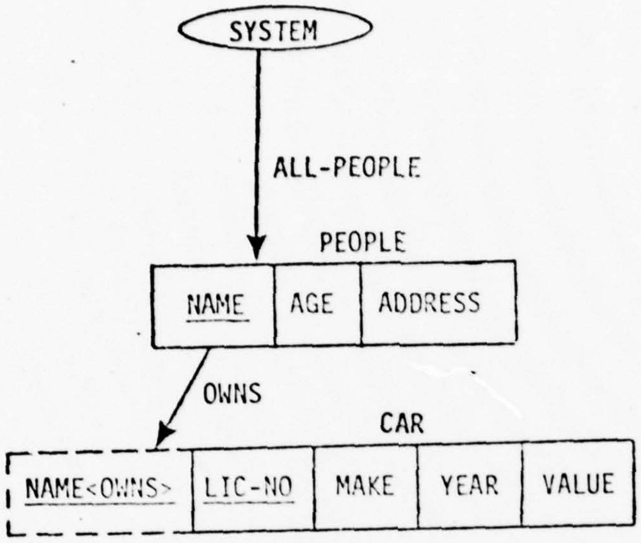

SYSTEM
ALL-PEOPLE
PEOPLE
NAME
AGE
ADDRESS
OWNS
CAR
NAME<OWNS>
LIC-NO
MAKE
YEAR
VALUE

Figure 3-6

```
1.  TARGET RECORD PEOPLE
2.      ACCESS PATH PEOPLE-BUILDER
3.          SOURCE RECORD NEIGHBOR ACCESS VIA ALL-NEIGHBORS
4.              ADDR ASSIGN TO ADDRESS
5.              SOURCE RECORD PEOPLE ACCESS VIA LIVE-THERE
6.                  NAME ASSIGN TO NAME
7.                  AGE ASSIGN TO AGE
```

From the relational point of view PEOPLE-BUILDER instructs the restructuring system to retrieve all NEIGHBOR tuples, and for each particular NEIGHBOR tuple, retrieve all the PEOPLE tuples whose DEED#<LIVE> field matches the NEIGHBOR's DEED# field (i.e., all PEOPLE related to the NEIGHBOR along LIVE-THERE). From each such NEIGHBOR-PEOPLE pair, ADDR from the NEIGHBOR, and NAME and AGE from the PEOPLE are to be projected onto ADDRESS, NAME, and AGE in the target PEOPLE record.

All projections are specified in this way, i.e., by ITEM statements giving the source-domain-to-target-domain correspondence, or by BLOCK ASSIGNMENT statements which specify multiple correspondences simultaneously. All joins which are represented by source sets are specified as above; i.e., by simply instructing the restructuring system to traverse the appropriate set(s). Joins not represented by source sets are specified by describing access to each of the relations to be joined, and using selection criteria to guarantee matching join fields. This is illustrated in Section 3.2.2.

From the network perspective, an ACCESS PATH describes a hierarchical substructure of the source data. Each node of the hierarchy is described by a SOURCE RECORD statement. The set by which it is to be reached from its parent node is specified in the ACCESS VIA clause. APSL regards all sets as two-way sets; passage from parent node to child node may be either a member-to-owner or owner-to-member access. Since RIM sets cannot have multiple owner or member types, the basic form of the SOURCE RECORD statement completely determines parent node, with two exceptions. First, there may by more than one node on the hierarchy containing an instance of the parent node record type. In this case, the FROM clause resolves the ambiguity. Second, if the set has the same owner and member record type, the direction of access is not determined. The $\left\{\begin{matrix}\text{OWNER/MEMBER}\\\text{MEMBER/OWNER}\end{matrix}\right\}$ must be given. During restructuring, each instance of the hierarchy is located, and a target record instance is created from it with item values as specified by the ITEM statements on the ACCESS PATH.

3.2.2 Selections, More Joins

Figure 3-3 shows a YOUNG-BOSS relation, which is computed by joining the source relations PEOPLE and BOSS on NAME, and for all such tuples with the AGE <30, projecting NAME, AGE, and COMPANY into a target tuple.

The desired join is not represented explicitly in the source data, so access directions for BOSS and PEOPLE are given, and the selection on line 7 checks to see that a YOUNG-BOSS is created only from a BOSS and a PEOPLE with matching NAMES.

We adopt the following abbreviations:

TR	for TARGET RECORD
AP	for ACCESS PATH
SR	for SOURCE RECORD
AV	for ACCESS VIA
AT	for ASSIGN TO

```
1.  TR YOUNG-BOSS
2.      AP YOUNG-ONE
3.          SR BOSS AV ALL-BOSSES
4.              COMPANY ASSIGN TO COMPANY
5.          SR NEIGHBOR AV ALL-NEIGHBORS
6.              SR PEOPLE AV LIVE-THERE
7.                  NAME SELECT IF EQ NAME FROM BOSS AT NAME
8.                  AGE SELECT IF LT 30 AT AGE
```

A target record (or tuple) is created form an instance of a source hierarchy (or set of source tuples) only if all the selection criteria specified for the ACCESS PATH are satisfied.

3.2.3 Unions

The relational algebra operation of Union is easily specified in APSL; a target relation is the union of all tuples created from all the ACCESS PATHs specified for that relation. For example, in Figure 3-4 the POOR-SISTER relation is the union of POOR-SISTERs related through the mother and those through the father. The two possibilities are accounted for by the POOR-MOM and POOR-DAD ACCESS PATHS respectively.

relational algebra. In addition, a synthesis of the network and relational perspectives is possible. To some extent, the user has the "best of both worlds"; he can select the data model best suited to a particular portion of his restructuring transformation.

7

```
TR PEOPLE
    AP PEOPLE
    SR NEIGHBOR AV ALL-NEIGHBORS
        SR PEOPLE AV LIVE-THERE
            NAME AT NAME
            AGE AT AGE
            NET-WORTH AT NET-WORTH
TR POOR-SISTER
    AP POOR-MOM
        SR NEIGHBOR AV ALL-NEIGHBORS
            SR PEOPLE ID=MOM AV LIVE-THERE
            SR PEOPLE ID-KID AV MOTHER-OF FROM MOM OWNER/MEMBER
                NAME AT NAME<POOR>
            SR PEOPLE ID=SIS AV MOTHER-OF FROM MOM OWNER/MEMBER
                NAME SELECT IF NE NAME FROM PEOPLE ID=KID AT NAME
                NET-WORTH SELECT IF LE 2000 AT NET-WORTH
                SEX SELECT IF EQ 'FEMALE'
    AP POOR-DAD
        SR NEIGHBOR AV ALL-NEIGHBORS
            SR PEOPLE ID=DAD AV LIVE-THERE
                SR PEOPLE ID=KID AV FATHER-OF FROM DAD OWNER/MEMBER
                    NAME AT NAME<POOR>
                SR PEOPLE ID=SIS AV FATHER-OF FROM DAD OWNER/MEMBER
                    NAME SELECT IF NE NAME FROM PEOPLE ID=KID AT NAME
                    NET-WORTH SELECT IF LE 2000 AT NET-WORTH
                    SEX SELECT IF EQ 'FEMALE'
```

3.2.4 Network-Oriented Operations

Suppose performance considerations require frequent fast access from a NEIGHBOR record to all the AUTOs which belong to PEOPLE who live in the house, and suppose that no car is jointly owned by PEOPLE living in different houses. Then the CARS set of Figure 3-5 is a possible solution. It could be implemented as follows:

```
TR NEIGHBOR
    AP NEIGH
        SR NEIGHBOR AV ALL-NEIGHBORS
            ACTUAL DATA IN ORDER
TR AUTO
    AP AUTO-BUILDER
        SR NEIGHBOR AV ALL-NEIGHBORS
            DEED# AT DEED#<CARS>
            SR PEOPLE AV LIVE-THERE
                SR POSSESSION-LINK AV OWNS
                    SR AUTO AV OWNED-BY
                        ACTUAL DATA IN ORDER
```

Another frequently occurring network restructuring transformation involves changing the implementation of many-to-many relationships. In Figure 3-6, ownership of cars is represented by an AUTO record for each owner. This may involve duplicate AUTO data, of course. We will assume that PEOPLE is created according to the PEOPLE-BUILDER ACCESS PATH of Section 3.2.1.

```
TR CAR
    AP CAR-FACTORY
        SR AUTO AV ALL-CARS
            ACTUAL DATA IN ORDER
            SR POSSESSION-LINK AV OWNED-BY
                NAME<OWNS>AT NAME<OWNS>
```

3.2.5 Partial Restructuring

Suppose that the ROOM, LAMP, and FURNITURE record types are not to be changed between source and target. Then APSL allows rather simple specifications for them:

```
TR ROOM IS SR ROOM
TR LAMP IS SR LAMP
TR FURNITURE IS SR FURNITURE
```

Partial restructuring specifications may be somewhat more complex when sets connect the unchanging portion of the database to the restructured portion. (See appendix).

4.0 CONCLUSIONS

We have presented the salient features of an approach to network database restructuring which we feel has considerable power and generality. These include the RIM data model, which permits a database to be viewed simultaneously as a network and as a relational database in first normal form, and a restructuring system which can perform all the operations of the relational algebra, but is not restricted to any set of data-model-dependent operations. We have described a powerful, but structurally simple language, APSL, for the specification of restructuring transformations, and we have illustrated its use in describing both relational- and network-oriented operations. APSL has been implemented as part of the Michigan Data Translator [DT].

Finally, it is important to observe that although APSL was developed as a restructuring specifications language, it has potential applications outside the restructuring area. For example, since APSL descriptions are not based on sequences of operations which manipulate the constructs of a particular data model, it may prove useful in specifying high-level DML operations in the area of database application program description, validation, and translation.

APPENDIX A
A DETAILED EXAMPLE.

Figure A-1 shows a Bachman diagram for a database schema, augmented to include set-significant items (in dotted boxes) and primary key items (underlined). For ease of reference, Figure A-2 shows its records and sets. The database is a thoroughly contrived one which might be used as an aid in the upkeep of a large mansion. Its rather bizarre semantics are summarized.

1.	ROOM	one for each room of the mansion, including the three garages. Rooms are identified by their names. Also stored are the rooms' dimensions and functions.
2.	CLOSET	one for each closet in the house. A closet number is unique over all closets in the mansion. Also known are the closet's location (i.e., north wall, west wall, etc.), dimensions, and door type (i.e., sliding, folding, swinging, etc.).
3.	VALUABLES	one for each object worth at least $500 currently stored in a closet. Also given are a description of the valuable (TYPE) and its (appraised) VALUE.
4.	POSSESSION-LINK	a link record used to represent ownership (perhaps joint) of valuables and vehicles.
5.	PEOPLE	one for each resident of the mansion and each neighbor. Roles of items should be apparent.
6.	NEIGHBORS	one for each neighboring household (the area is zoned for one-family dwellings, so this is equivalent to one for each nearby house). Roles of items should be apparent.
7.	PRIME-USER	a link record used to represent who uses which room most frequently. The cook will be the prime user of the kitchen, the aging patriarch the prime user of the study, etc. Some rooms, such as the dining room, living room, and billiard room will have several prime users. Neighbors are forbidden to be prime users of rooms. Thus, a PEOPLE record which owns a USES set cannot be a member of LIVES-THERE and vice versa.

8.	AUTO	one for each car owned by mansion residents or neighbors. Roles of items should be apparent. AUTO records for cars belonging to mansion residents are members of the CARS set owned by the garage in which they are kept.
9.	LOCATION-LINK	a link record used to establish past and present locations of the mansion's furnishings (LIGHTING, FURNISHING, FLOORS, and WINDOWS are mutually exclusive sets). Current-flag is 1 if the object is now in the ROOM, 0 otherwise. Last-date has no meaning if Current-flag is 1. If current-flag is 0, Last-date is the date the object was last moved out of the ROOM.
10.	LAMP	one for each piece of lighting equipment in the mansion. Burn-flag indicates whether or not bulb(s) in the lamp are burned out.
11.	FURNITURE	one for each piece of furniture in the mansion. Repair flag is 1 if the object needs repair, 9 if not. If Repair-flag = 1, Repair-Desc contains a description (in 256 characters or less) of the needed repairs.
12.	CARPETING	one for each piece of floor covering in the house. Mend-flag and Mending-Desc work the same way as Repair-flag and Repair-Desc for furniture.
13.	WINDOW	one for each of the mansion's windows. They are identified by window numbers taken from the mansion's blueprints, since insurance numbers were not assigned. Open-flag indicates whether or not the window can be opened, storm-flag whether or not it takes a storm window, and breakage-flag whether or not the window is currently broken. If the window is broken and a neighbor is at fault, CULPRIT contains the ADDR of the household to which a bill will be sent. CULPRIT is blank if the guilty party is unknown, not human, or a resident of the mansion. Observe that members of the WINDOWS set always have Current-flag = 1.

The bulk of the database is taken up by a great many of the large LAMP, FURNITURE, CARPETING, and WINDOW records. These are to be left unchanged by a restructuring transformation whose primary purpose is to separate the residents of the mansion from their neighbors (US and THEM, respectively). Figure A-3 shows the desired target database; Figure A-4, its records and sets. Several other changes are to be made as well, notably the addition of OLD-RICH-UNCLE records. An instance of the BUCKS set consists of an US record as owner, and as members, one OLD-RICH-UNCLE record for each of that person's uncles who is at least 65 and worth at least $500,000.

APSL statements describing this restructuring transformation, together with some explanatory comments, make up the remainder of this appendix.

The bulk of the database is taken up by a great many of the large LAMP, FURNITURE, CARPETING, and WINDOW records. These are to be left unchanged by a restructuring transformation whose primary purpose is to separate the residents of the mansion from their neighbors (US and THEM, respectively). Figure A-3 shows the desired target database; Figure A-4, its records and sets. Several other changes are to be made as well, notably the addition of OLD-RICH-UNCLE records. An instance of the BUCKS set consists of an US record as owner, and as members, one OLD-RICH-UNCLE record for each of that person's uncles who is at least 65 and worth at least $500,000.

APSL statements describing this restructuring transformation, together with some explanatory comments, make up the remainder of this appendix.

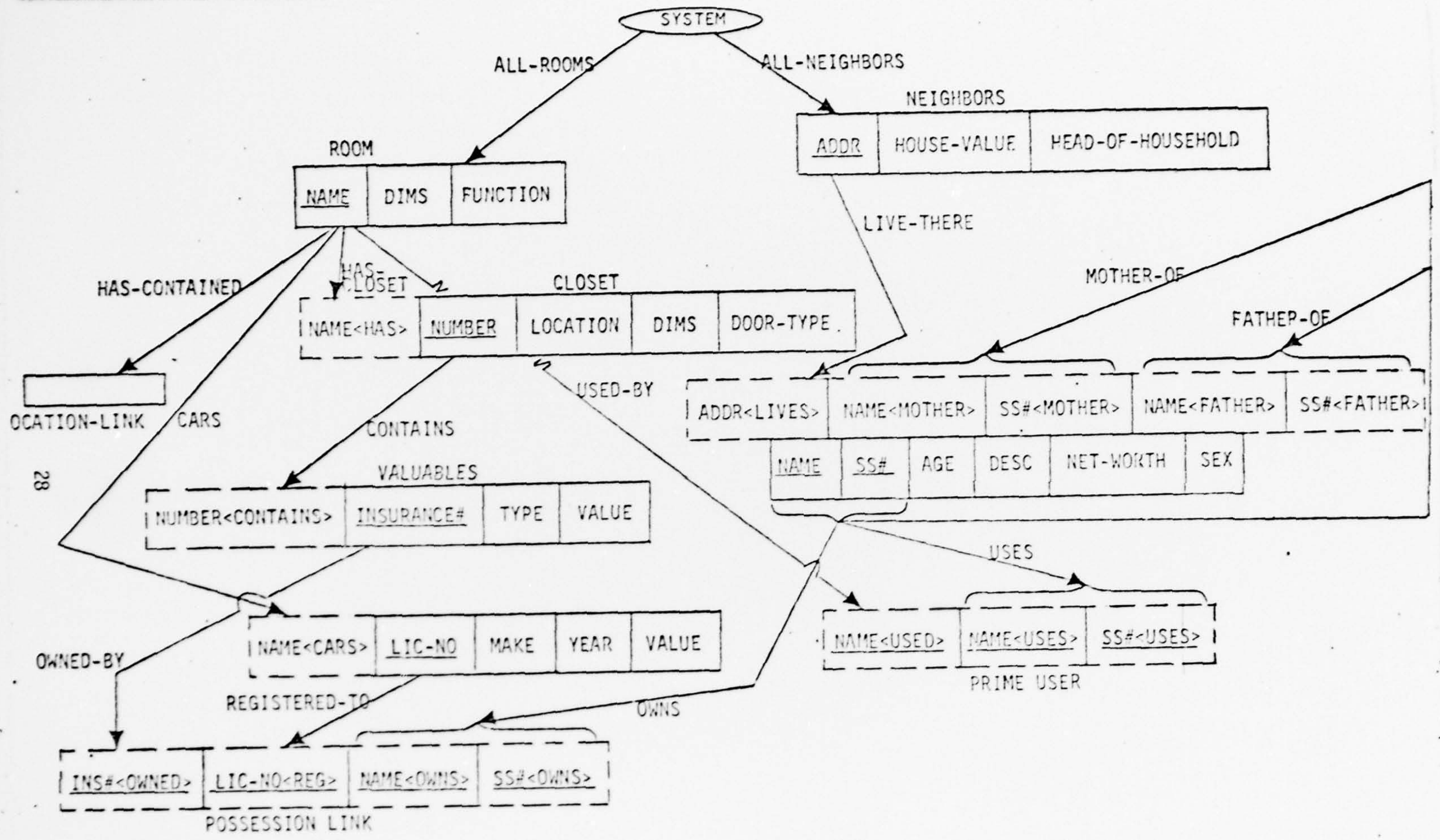


Figure A-1
Mansion Database (Source Only)

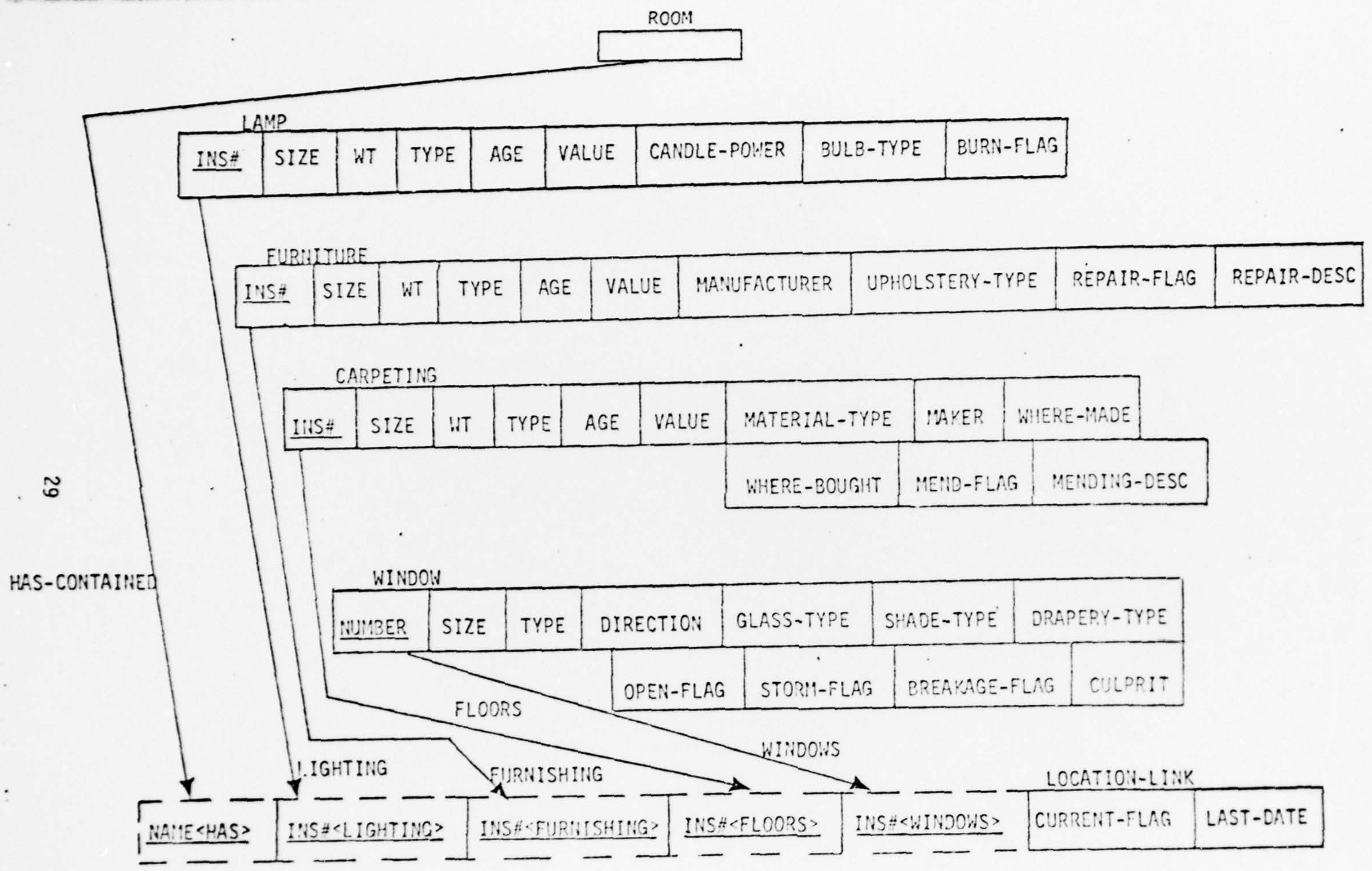


Figure A-1 (continued)
Mansion Database (Unchanging Portion)

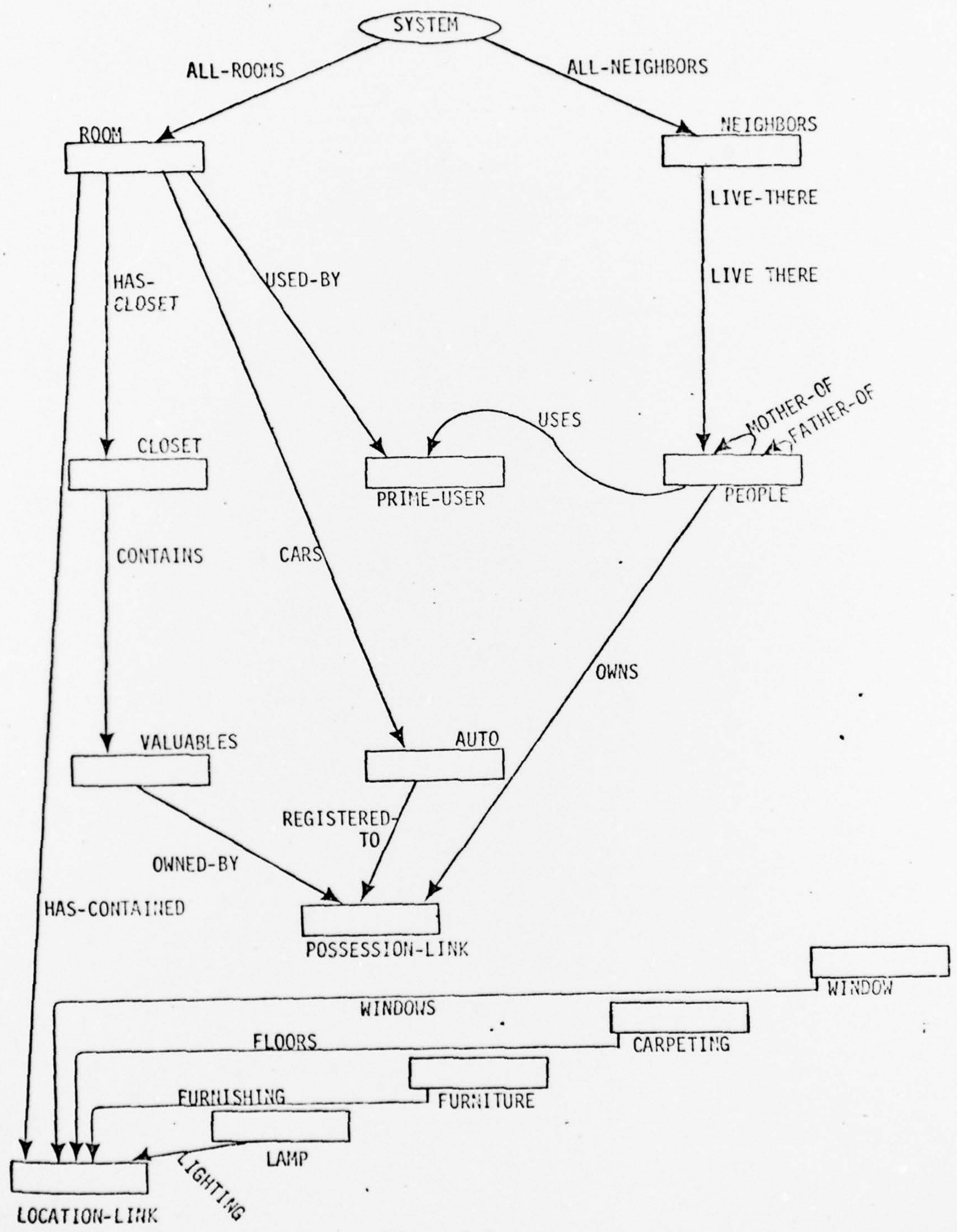


Figure A-2
Source Records and Sets

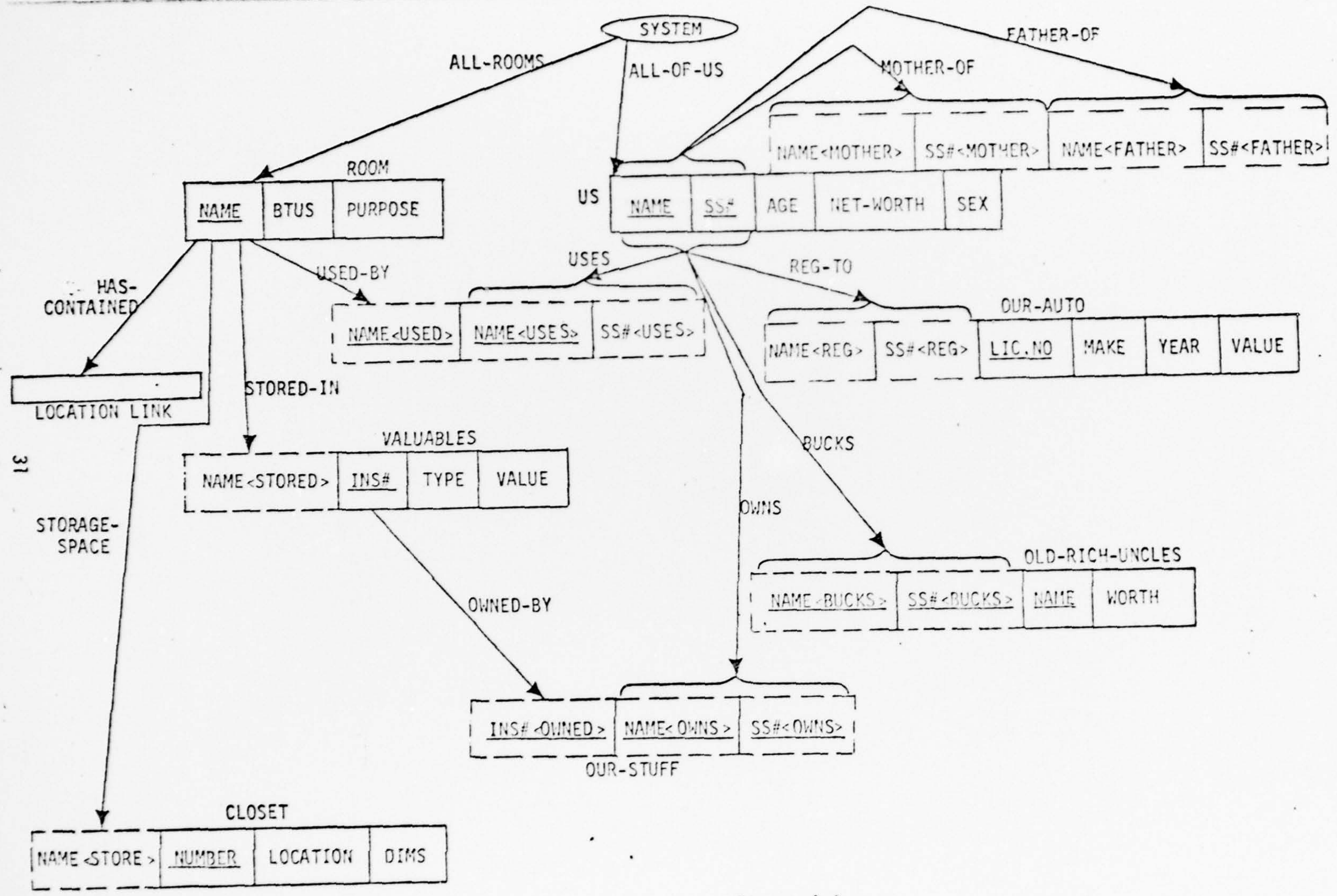


Figure A-3
Mansion Database (Target Only-residents' portion)

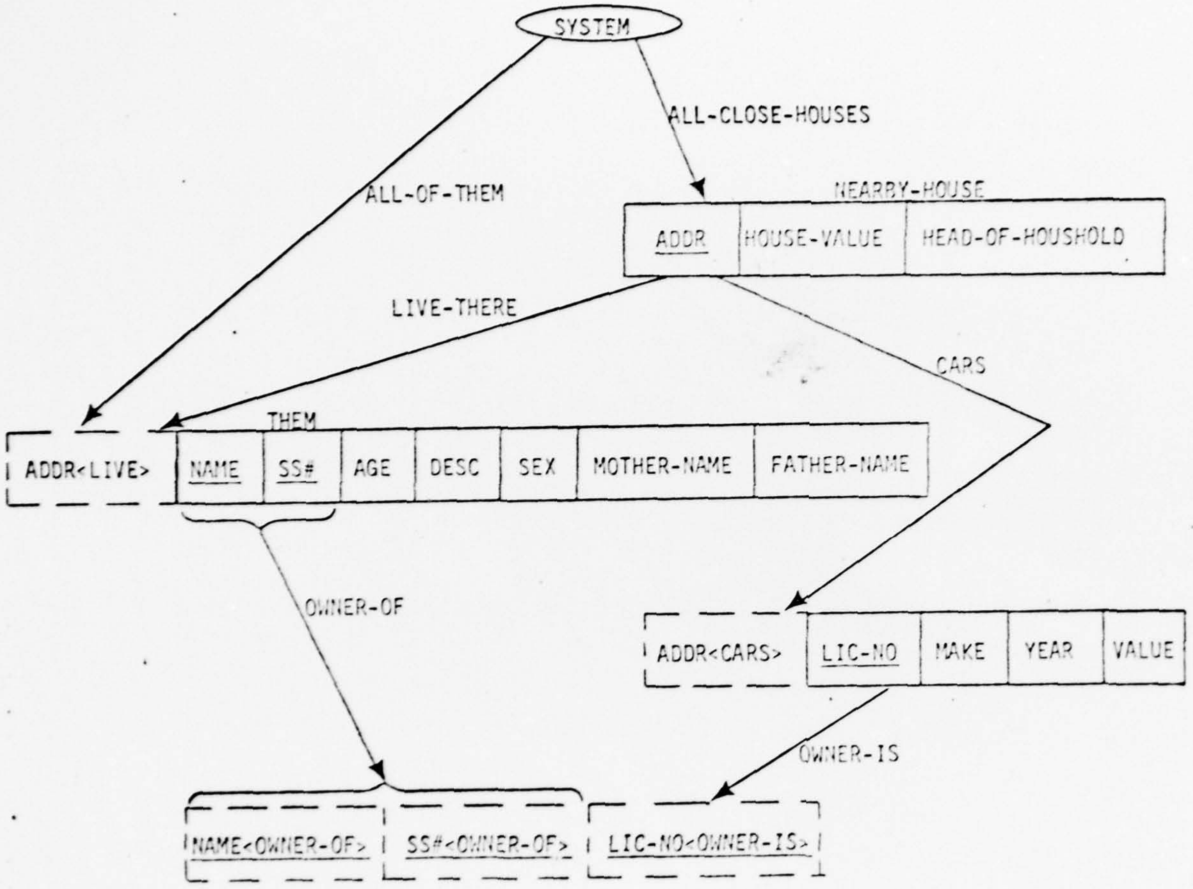


Figure A-3 (continued)
Mansion Database (Target Only - Neighbors' Portion)

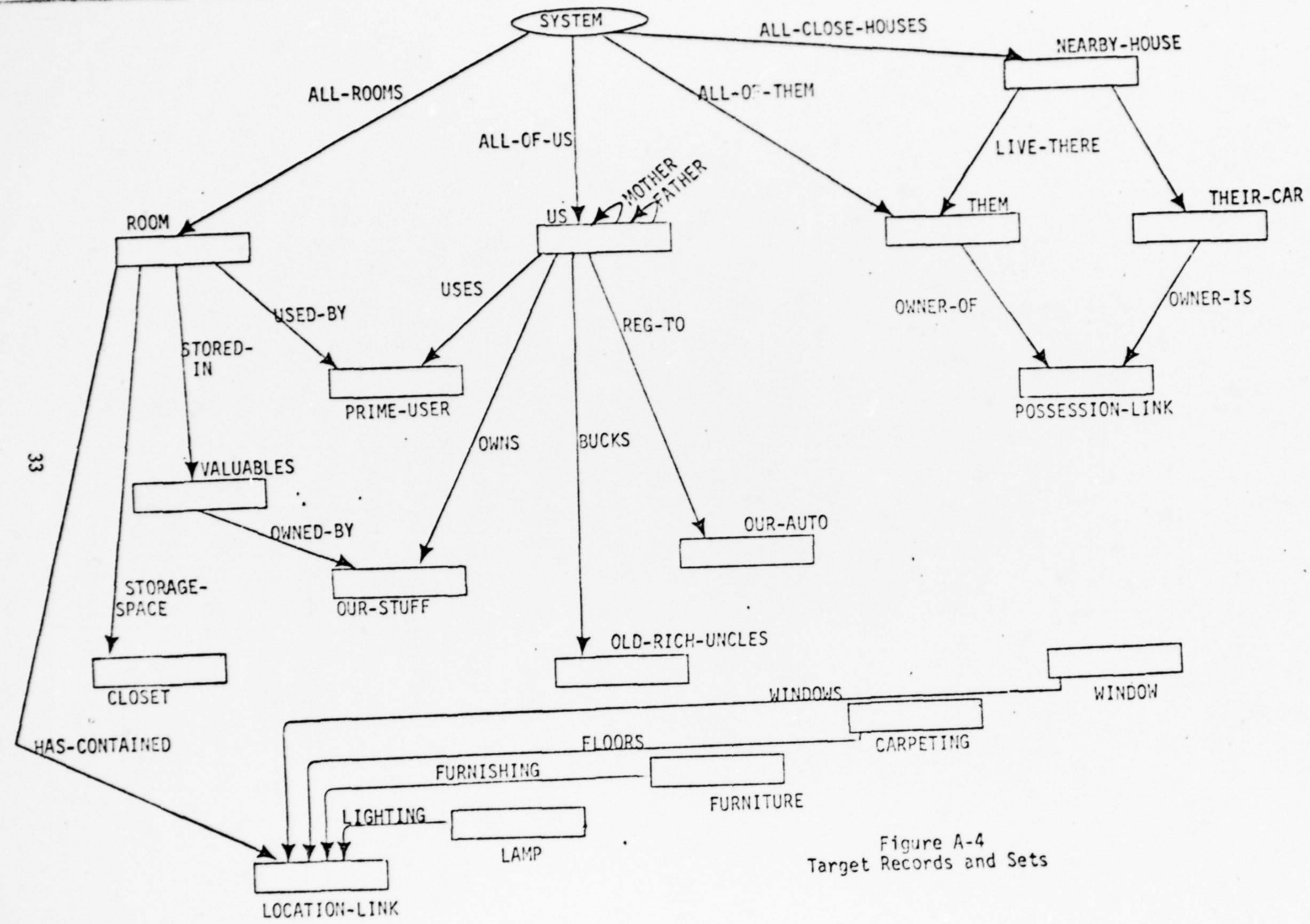


Figure A-4
Target Records and Sets

```
0.  /* MANSION DATABASE RESTRUCTURING APSL DESCRIPTION */
1.  MACRO SR LITERALLY 'SOURCE RECORD'
2.  MACRO TR LITERALLY 'TARGET RECORD'
3.  MACRO AV LITERALLY 'ACCESS VIA'
4.  MACRO AT LITERALLY 'ASSIGN TO'
4.5 MACRO AP LITERALLY 'ACCESS PATH'
```

Remarks

--Comments begin with /* and end with */

--APSL macro facility is limited to the form MACRO <Word> LITERALLY

'<APSL Reserved Word>'

```
5.  /* FIRST, THE UNCHANGING PORTION */
6.  TARGET RECORD LOCATION-LINK IS SOURCE RECORD LOCATION-LINK
7.  SET HAS-CONTAINED IS HAS-CONTAINED
8.     NAME<HAS-CON> IS NAME<HAS-CON>
9.  TARGET RECORD LAMP IS SOURCE RECORD LAMP
10. TARGET RECORD FURNITURE IS SOURCE RECORD FURNITURE
11. TARGET RECORD CARPETING IS SOURCE RECORD CARPETING
12. TARGET RECORD WINDOW IS SOURCE RECORD WINDOW
```

Remarks

--The SET...specification of lines 7 and 8 is required because HAS-CONTAINED is owned by the restructured record type ROOM.

```
13. TR ROOM
14.    ROOMIE
15.       SR ROOM AV ALL-ROOMS
16.          NAME AT NAME
17.           FUNCTION AT FUNCTION
18.            DIMS AT BTUS CONVERT WITH HEAT
```

Remarks

--HEAT is the name of a routine which accepts the dimensions of a room as input and produces the number of BTUs required to heat the room 1° C in 1 minute.

```
19. TR CLOSET
20.   AP CLOSET
21.       SR ROOM AV ALL-ROOMS
22.          NAME AT NAME<STORE>
23.          SR CLOSET AV HAS-CLOS
24.             NUMBER AT NUMBER
25.             LOCATION AT LOCATION
26.             DIMS AT DIMS
```

Remarks

--From the relational viewpoint, this AP describes the projection of the RIM relation
CLOSET (NAME<HAS>, NUMBER, LOCATION, DIMS, DOOR-TYPE)
onto its first four fields.

```
27.  TR VALUABLES
28.    AP LUX
29.        SR ROOM AV ALL-ROOMS
30.          NAME AT NAME<STORED>
31.          SR CLOSET AV HAS-CLOS
32.             SR VALUABLES AV CONTAINS
33.             ACTUAL DATA IN ORDER
```

Remarks

--From the relational viewpoint, this AP describes the join of CLOSET (NAME<HAS>, NUMBER, LOCATION, DIMS, DOOR-TYPE) and VALUABLES (NUMBER<CONTAINS>, INSURANCE#, TYPE, VALUE) on the NUMBER fields, followed by projection onto the NAME<HAS>, INSURANCE#, TYPE, and VALUE fields.

```
34.  TR OUR-STUFF
35.    AP OURS
36.       SR ROOM AV ALL-ROOMS
37.          SR  CLOSET AV HAS-CLOS
38.              SR VALUABLES AV CONTAINS
39.                 INSURANCE# AT INS#<OWNED>
40.                    SR POSSESSION-LINK AV OWNED-BY
41.                      NAME<OWNS> AT NAME<OWNS>
42.                      SS#<OWNS> AT SS#<OWNS>




43.  TR PRIME-USER
44.    AP PRIME
45.       SR ROOM AV ALL-ROOMS
46.         SR PRIME-USER AV USED-BY
47.           ALL DATA BY NAME
48  TR US
49.    AP WE
50.       SR ROOM AV ALL-ROOMS
51.         SR PRIME-USER AV USED-BY
52.           SR PEOPLE ID=PERSON AV USES
53.              NAME AT NAME
54.              SS# AT SS#
55.              AGE AT AGE
56.              NET-WORTH AT NET-WORTH
57.              SEX AT SEX
58.              SR PEOPLE ID=MOM AV MOTHER-OF
                          FROM PERSON MEMBER/OWNER
59.                 ACCEPT IF NULL
60.                 NAME AT NAME<MOTHER>
61.                 SS# AT SS#<MOTHER> NULL VALUE = 0
62.              SR PEOPLE ID=DAD AV FATHER-OF
                          FROM PERSON MEMBER/OWNER
63.                 ACCEPT IF NULL
64.                 NAME AT NAME<FATHER>
65.                 SS# AT SS#<FATHER>NULL VALUE=0
```

Remarks

--Observe the use of identifiers in distinguishing anong the three appearances of PEOPLE on this tree.

--Since every resident of the mansion is the prime user of a bedroom, and only residents of the mansion can be prime users of rroms, this AP completely describes the construction of target US records.

--Note further that since a person may be a prime user of several rooms, a restructuring system driven by this APSL description will create and discard some duplicate US record instances.

--ACCEPT IF NULL is used to prevent the loss of PEOPLE records which are not members of the FATHER-OF and/or MOTHER-OF sets.

--MEMBER/OWNER is used to guarantee that a person's parents, rather than his or her children, are recorded as MOTHER and FATHER.

```
 /* THEM RECORDS ARE SIMILAR TO US RECORDS */
 /* EXCEPT THAT THEY ARE FOUND IN THE SOURCE */
 /* BY PASSING THROUGH NEIGHBORS */
TR THEM
  AP THEY
     SR NEIGHBORS AV ALL-NEIGHBORS
       ADDR AT ADDR<LIVE>
       SR PEOPLE ID=PERSON AV LIVE-THERE
         NAME AT NAME
         SS# AT SS#
         AGE AT AGE
         DESC AT DESC
         SEX AT SEX
         SR PEOPLE ID=MOM AV MOTHER-OF
                   FROM PERSON MEMBER/OWNER
            ACCEPT IF NULL
            NAME AT MOTHER-NAME
                    NULL VALUE = '*UNKNOWN*'
         SR PEOPLE ID=DAD AV FATHER-OF
                   FROM PERSON MEMBER/OWNER
            ACCEPT IF NULL
            NAME AT FATHER-NAME
                   NULL VALUE = '*UNKNOWN*'
TR NEARBY-HOUSE
  AP NEAR
     ST NEIGHBORS AV ALL-NEIGHBORS
        ACTUAL DATA IN ORDER
 /* NOW FOR THE CARS */
  TR OUR-AUTO
    AP OURCAR
```

```
90.            SR ROOM AV ALL-ROOMS
91.              SR AUTO AV CARS
92.                ACTUAL DATA IN ORDER
93.                SR POSSESSION-LINK AV REGISTERED-TO
94.                  NAME<OWNS> AT NAME<REG-TO>
95.                  SS#<OWNS> AT SS#<REG-TO>
```

Remarks

--Observe that the choice of primary key for OUR-AUTO implies that residents of the mansion do not share ownership of their cars.

```
96.  TR THEIR-CAR
97.   AP THEIR CAR
98.        SR NEIGHBORS AV ALL-NEIGHBORS
99.           ADDR AT ADDR<CARS>
100.          SR PEOPLE AV LIVE-THERE
101.            SR POSSESSION-LINK AV OWNS
102.              SR AUTO AV REGISTERED-TO
103.                 ACTUAL DATA IN ORDER
104. TR POSSESSION-LINK
105.   AP POSSE
106.       SR NEIGHBORS AV ALL-NEIGHBORS
107.         SR PEOPLE AV LIVE-THERE
108.           SR POSSESSION-LINK AV OWNS
109.              LIC-NO<REG> AT LIC-NO<OWNER-IS>
110.              NAME<OWNS> AT NAME<OWNER-OF>
111.              SS#<OWNS>AT SS#<OWNER-OF>
112. /* LAST BUT CERTAINLY NOT LEAST, THE OLD */
113. /* RICH UNCLES */
114. TR OLD-RICH-UNCLES
115.   AP ORUNCMM
116.       SR ROOM AV ALL-ROOMS
117.         FUNCITON SELECT IF EQ 'BEDROOM'
118.         SR PRIME-USER AV USED-BY
119.           SR PEOPLE ID=GRANDMA AV USES
120.             SR PEOPLE ID=UNC AV MOTHER-OF
                          FROM GRANDMA OWNER/MEMBER
121.               SEX SELECT IF EQ 'MALE'
122.               AGE SELECT IF GE 65
123.               NET-WORTH SELECT IF GE 500000 AT WORTH
124.               NAME AT NAME
125.             SR PEOPLE ID=MOM AV MOTHER-OF
                          FROM GRANDMA OWNER/MEMBER
126.               SR PEOPLE ID=KID AV MOTHER-OF
                          FROM MOM OWNER/MEMBER
127.                  NAME AT NAME<BUCKS>
128.                  SS# AT SS#<BUCKS>
129.  AP ORUNCMF
130.      SR ROOM AV ALL-ROOMS
```

```
131.          FUNCTION SELECT IF EQ 'BEDROOM'
132.          SR PRIME-USER AV USED-BY
133.            SR PEOPLE ID=GRANDMA AV USES
134.              SR PEOPLE ID=UNC AV MOTHER-OF
                     FROM GRANDMA OWNER/MEMBER
135.                 SEX SELECT IF EQ 'MALE'
136.                 AGE SELECT IF GE 65
137.                 NET-WORTH SELECT IF GE 500000
                               AT WORTH
138.                 NAME AT NAME
139.              SR PEOPLE ID=DAD AV MOTHER-OF
                     FROM GRANDMA OWNER/MEMBER
140.                 NAME SELECT IF NE NAME
                         FROM PEOPLE ID=UNC
141.                 SR PEOPLE ID=KID AV FATHER-OF
                         FROM DAD OWNER/MEMBER
142.                     NAME AT NAME<BUCKS>
143.                     SS# AT SS#<BUCKS>
144.    AP ORUNC FF
145.       SR ROOM AV ALL-ROOMS
146.         FUNCTION SELECT IF EQ 'BEDROOM'
147.         SR PRIME-USER AV USED-BY
148.           SR PEOPLE ID=GRANDPA AV USES
149.             SR PEOPLE ID=UNC AV FATHER-OF
                    FROM GRANDPA OWNER/MEMBER
150.                SEX SELECT IF EQ 'MALE'
151.                AGE SELECT IF GE 65
152.                NET-WORTH SELECT IF GE 500000
                         AT WORTH
153.                NAME AT NAME
154.             SR PEOPLE ID=DAD AV FATHER-OF
                      FROM GRANDPA OWNER/MEMBER
155.                NAME SELECT IF NE NAME FROM
                           PEOPLE ID=UNC
156.                SR PEOPLE ID=KID AV FATHER-OF
                           FROM DAD OWNER/MEMBER
157.                   NAME AT NAME<BUCKS>
158.                   SS# AT SS#<BUCKS>
159.    AP ORUNC FM
160.        SR ROOM AV ALL-ROOMS
161.          FUNCTION SELECT IF EQ 'BEDROOM'
162.          SR PRIME-USER AV USED-BY
163.             SR PEOPLE ID=GRANDPA AV USES
164.               SR PEOPLE ID=UNC AV FATHER OF
                        FROM GRANDPA OWNER/MEMBER
165.                  SEX SELECT IF EQ 'MALE'
166.                  AGE SELECT IF GE 65
167.                  NET-WORTH SELECT IF GE 500000 AT WORTH
168.                  NAME AT NAME
169.             SR PEOPLE ID=MOM AV FATHER-OF
                      FROM GRANDPA OWNER/MEMBER
170.                  SR PEOPLE ID=KID AV MOTHER-OF
                         FROM MOM OWNER/MEMBER
171.                     NAME AT NAME<BUCKS>
172.                     SS# AT SS#<BUCKS>
173./* APSL DESCRIPTION COMPLETE */
```

References

DL28 NOVAK, D. and FRY, J., "The State of the Art of Logical Database Design", Proc. Fifth Texas Conference on Computing Systems, Austin, Texas, October 1976, pp.30-39.

DT1 FRY, J.P., FRANK, R.L., and HERSHEY, E.A., III, "A Developmental Model for Translation", Proc. 1972 ACM SIGFIDET Workshop on Data Description, Access and Control, Denver, Colorado, November 1972, pp.77-106.

DT3 MERTEN, A.G., and FRY, J.P., "A Data Description Approach to File Translation", Proc. 1974 ACM SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Michigan, May 1974, pp.191-205.

DT4 HOUSEL, B., LUM, V., and SHU, N., "Architecture to an Interactive Migration Systems (AIMS)", Proc. 1974 ACM SIGFIDET Workshop on Data Description, Access and Control, Ann Arbor, Michigan, May 1974, pp.157-169.

DT9 BAKKOM, D.E. and BEHYMER, J.A., "Implementation of a Prototype Generalized File Translator", Proc. 1975 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 1975, pp.99-110.

DT10 SHOSHANI, A., "A Logical Level Approach to the Data Base Conversion", Proc. 1975 ACM SIGMOD Conference, San Jose, California, May 1975, pp.112-122.

DT11 BIRSS, E.W., and FRY, J.P. "Generalized Software for Translating Data", Proc. 1976 NCC, Vol. 45. AFIPS Press, Montvale, New Jersey, pp.889-899.

DT12 BODWIN, JAMES, et al., "Data Translator Version IIA Release 2 User Manual", Technical Report 76 DT 3.4, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, forthcoming.

R2 NAVATHE, S.B., and FRY, J.P. "Restructuring for Large Data Bases", ACM Transactions on Database Systems 1,2 (June 1976) ACM, New York, 1976, pp.138-158.

R3 SHU, N.C., HOUSEL, B.C., and LUM, V.Y. "CONVERT: A High-Level Translation Definition Language for Data Conversion", Comm. ACM 18,10, October 1975, pp.57-67.

R5 SHOSHANI, A., "A Logical-Level Appraoch to Data Base Conversion", Proc. ACM/SIGMOD International Conference on Management of Data, ACM, New York, 1975.

R7 DEPPE, M.E., "A Relational Interface Model for Database Restructuring", Technical Report 76 DT 3, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1975.

R9 DEPPE, M.E., and LEWIS, K.H., "Data Translation Translation Definition Language Reference Manual for Version IIA Translator Release 1" Working Paper DT 5.2, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

M14 KERSHBERG, L., KLUG, A., and TSICHRITZIS, D., "A Taxonomy of Data Models", TR CSRG-70, Computer System Research Group, University of Toronto, May 1976.